Divide and Conquer: Neuroevolution for Multiclass Classification

Tyler McDonnell
SparkCognition, Inc. in Austin, TX
The University of Texas at Austin in Austin, TX
tmcdonnell@sparkcognition.com

Sari Andoni, Elmira Bonab, Sheila Cheng, Jun-Hwan Choi, Jimmie Goode, Keith Moore, Gavin Sellers
SparkCognition, Inc. in Austin, TX

Jacob Schrum
SparkCognition, Inc. in Austin, TX
Southwestern University in Georgetown, TX
jschrum@sparkcognition.com

ABSTRACT

Neuroevolution is a powerful and general technique for evolving the structure and weights of artificial neural networks. Though neuroevolutionary approaches such as NeuroEvolution of Augmenting Topologies (NEAT) have been successfully applied to various problems including classification, regression, and reinforcement learning problems, little work has explored application of these techniques to larger-scale multiclass classification problems. In this paper, NEAT is evaluated in several multiclass classification problems, and then extended via two ensemble approaches: One-vs-All and One-vs-One. These approaches decompose multiclass classification problems into a set of binary classification problems, in which each binary problem is solved by an instance of NEAT. These ensemble models exhibit reduced variance and increasingly superior accuracy as the number of classes increases. Additionally, higher accuracy is achieved early in training, even when artificially constrained for the sake of fair comparison with standard NEAT. However, because the approach can be trivially distributed, it can be applied quickly at large scale to solve real problems. In fact, these approaches are incorporated into Darwin™, an enterprise automatic machine learning solution that also incorporates various other algorithmic enhancements to NEAT. The resulting complete system has proven robust to a wide variety of client datasets.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; **Ensemble methods**; *Supervised learning*;

KEYWORDS

Neuroevolution, Ensembling, Neural Networks

ACM Reference Format:
Tyler McDonnell, Sari Andoni, Elmira Bonab, Sheila Cheng, Jun-Hwan Choi, Jimmie Goode, Keith Moore, Gavin Sellers, and Jacob Schrum. 2018. Divide and Conquer: Neuroevolution for Multiclass Classification. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205476

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205476

1 INTRODUCTION

Neuroevolution is a powerful and general technique for evolving the weights and, in some cases, structure of artificial neural networks. The ability to evolve the structure of the network remains a major benefit of evolutionary approaches over optimization techniques such as backpropagation, which tune a fixed number of parameters.

A popular approach to evolving the weights and structure of neural networks is NeuroEvolution of Augmenting Topologies (NEAT [25]). Variants of NEAT have been successful at regression [14], classification [4, 14], and reinforcement learning (RL) [17, 23, 24]. Because NEAT evolves networks from a simple starting point, it generally develops the minimal structure needed to address a problem, which helps with generalization and model understandability.

However, when a complicated model is required, especially one involving many output neurons, NEAT often struggles to optimize all parts of the model. In particular, multiclass classification problems can be challenging for NEAT. However, ensemble approaches can break up complex problems into manageable subproblems [11].

Ensembles leverage multiple trained models to make a joint decision regarding whatever problem they face. This approach has been applied using a variety of machine learning models [3, 10, 11, 16]. Ensemble approaches to neuroevolution [1, 7, 12, 22, 27] often select ensemble members from a single evolving population. However, in the age of Big Compute, it is easy and effective to run multiple distinct evolutionary runs, each with its own population, to create specialist models for subproblems within a larger task.

The approach examined in this paper is the evolution of ensembles for multiclass classification problems using NEAT. Specifically, One-vs-All and One-vs-One ensembles [11] are evolved. Despite instantiating multiple evolutionary runs for a single problem, the approach is not only effective at improving final classification performance, but substantially more efficient in terms of time, even without parallelizing the independent evolutionary runs.

When runs are parallelized, results are even more impressive. In fact, the approach is so effective that that it has been incorporated into Darwin™[1], an enterprise automatic machine learning solution that uses neuroevolution and various other techniques, such as backpropogation and decision trees, to produce models for a variety of supervised, unsupervised, and RL problems. Because of the complexity of Darwin™, this paper focuses exclusively on the performance of standard NEAT within the domain of multiclass classification problems and develops a straightforward comparison of standard NEAT to two ensemble approaches which utilize NEAT. After presenting results that clearly demonstrate the benefit

[1] https://www.sparkcognition.com/darwin/

of these ensembling techniques applied to NEAT, the role of these techniques within Darwin™is discussed, along with a list of the many features of Darwin™that will be the topic of future research.

2 PREVIOUS WORK

Classification is a common supervised learning problem that has been addressed by a variety of machine learning approaches. Similarly, ensemble learning has been used with a variety of models to achieve results superior to what could generally be achieved with any one model. This section focuses on previous approaches to classification and ensemble learning that rely on evolution.

Because neural networks are theoretically capable of representing any continuous function on a hypercube [6], they have been applied to many problems in machine learning, including classification. However, despite their theoretical capabilities, training them is non-trivial. The most common approach to training neural networks is backpropagation. This approach has been used to train ensembles of neural networks for multiclass classification [2].

However, the approach used in this paper is neuroevolution: the evolution of neural networks. An early neuroevolution approach to classification was EPNet [26], which evolved the topologies of neural networks, but used a mixture of backpropagation and simulated annealing to train weights. This approach did not use crossover. EPNet performed reasonably well on simple classification problems.

A later neuroevolution algorithm is NEAT [25], which is described further in Section 3.1. Briefly, standard NEAT evolves the topologies and weights of its neural networks without backpropagation or any additional weight tuning algorithm. NEAT also allows for efficient meaningful crossover of network genomes. NEAT and its variants have been applied to classification problems in the past [4, 14], attaining good results for problems with up to three classes.

NEAT has been used to evolve ensembles [22], but this approach differs from ours in that it was applied to RL problems rather than supervised classification problems, and the ensemble members came from a single population. Our approach relies on multiple populations evolved in parallel, as described in Section 3.2.

In fact, several approaches to evolving neural network ensembles (not using NEAT) take the ensemble members from a single population. Yao and Liu evolved such an ensemble by training the individual networks with negative correlation learning [27], but only applied their method to binary classification problems. Abbass used Pareto-based multiobjective evolution to create a single population of neural networks trained by backpropagation, whose Pareto front was made into an ensemble [1]. There is even a cooperative coevolution approach by Garcia-Pedrajas et al. in which one population has individuals that identify the neural networks of another population that participate in an ensemble [12].

A related approach to neuroevolution of ensembles is Neural Learning Classifier systems [7], which are an evolutionary approach to discovering a population of rules that fire under specific circumstances. In this case, the rules are neural networks, which are also trained using negative correlation learning to create an ensemble.

The experiments in this paper show that a neuroevolution approach relying only on mutation to modify its weights can evolve several separate populations that combine into an effective ensemble. The specific methods used in this approach are described next.

3 METHODS

The NEAT neuroevolution approach is described, before describing two ensemble approaches that make use of it.

3.1 NEAT

NeuroEvolution of Augmenting Topologies (NEAT [25]) is an evolutionary algorithm that evolves artificial neural networks with arbitrary topologies. Networks start simple, but *complexify* over generations via structural mutations that splice new neurons along existing links, and add links between existing neurons. Networks are tuned via mutations that change the weights of existing links.

The implementation of NEAT in this paper also allows for mutations that delete links or nodes (and all connected links), and change activation functions. The default activation function for new nodes is hyperbolic tangent (tanh), but an activation function mutation can change any node's function to one from this set: tanh, sigmoid, ReLU, SELU, Gaussian, softsign, square, or identity.

Other important innovations of NEAT include efficient crossover of networks that are aligned via the tracking of historical markers, and the use of speciation and fitness sharing to protect innovation.

NEAT has been applied to a variety of problems, including regression [14], classification [4, 14], and RL problems [17, 23, 24]. As mentioned in Section 2, it has even been used to evolve ensembles [22], though differently from what is proposed in this paper.

When solving classification problems, every network in the evolving population is applied to every entry of a training data set. Each network has one output neuron for each possible class, and the output with the highest value indicates the network's classification. The fitness of the network is $1-CE$, where CE is the average cross entropy loss across all training examples, where the loss for one example is defined as $-\sum_{i=1}^{k} y_i \ln(p_i)$ for k classes where y_i is a binary indicator for whether i is the correct label for the current observation, and p_i is the network output corresponding to class i.

However, because networks have one output neuron per class, NEAT struggles with classification problems involving many classes. Tuning the behavior of each output neuron is difficult, because adequate network structure and suitable link weights must be evolved for each output. Ensemble approaches overcome this problem by dividing a multiclass classification task into sub-problems. The ensemble approaches used in this paper are described next.

3.2 Divide and Conquer

Two divide and conquer based approaches for evolving neural network ensembles are presented. Each method reduces a multiclass classification task into a collection of binary classification tasks and applies NEAT to each subtask to produce an ensemble which addresses the multiclass problem. Such binarization approaches have been widely deployed [10, 11, 16], as some learning techniques such as support vector machines were originally conceived to solve only binary classification problems [15]. Thus, these reduction techniques have historically provided a straightforward mechanism for extending binary classifiers to multiclass problems. Such approaches are also attractive because they are easily parallelized.

3.2.1 One-vs-All-NEAT. The first strategy explored is One-vs-All (OVA) multiclass decomposition [11]. Given a problem with k

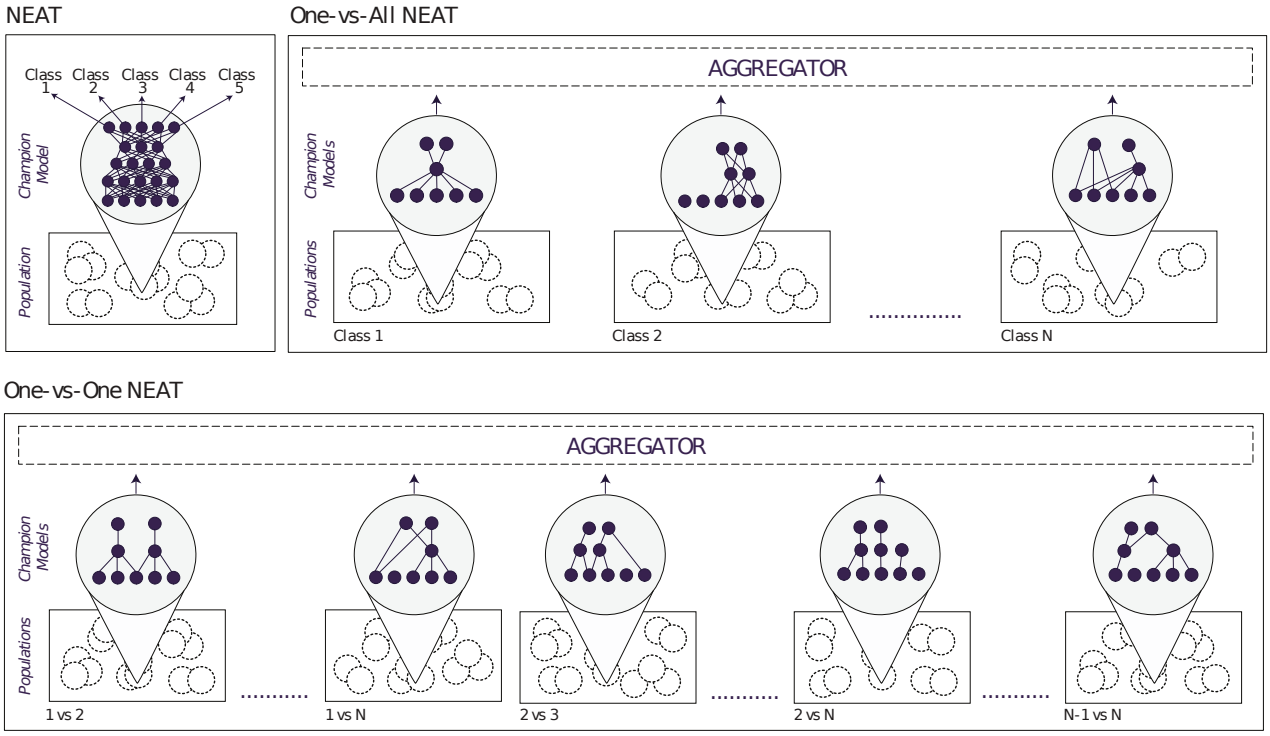

Figure 1: Divide and Conquer Methodologies: Standard NEAT compared with the OVA-NEAT and OVO-NEAT divide and conquer strategies, which decompose a multiclass problem into a set of related binary tasks and ensemble an array of NEAT champions. NEAT must evolve more complex individual champions to be successful, whereas ensemble approaches evolve teams of specialists that are superior despite their individual comparative simplicity. In principle, a variety of aggregation mechanisms can be used to determine the decisions of the ensembles, though the specific mechanisms used in this paper are described in sections 3.2.1 (OVA-NEAT) and 3.2.2 (OVO-NEAT).

classes, OVA generates k binary classifiers, where each classifier is trained to distinguish a single class from the remaining classes. We evolve each classifier using NEAT. Specifically, k instances of NEAT are initiated, with each instance corresponding to a single class. The goal of each classifier is to distinguish between samples which are a member of the class (positive samples) and those which are not (negative samples). Each classifier is trained using all training data, with labels modified to be -1 and +1 for negative and positive examples respectively. The final pool of classifiers consists of a corresponding champion from each binary classification problem.

To address the original multiclass problem, a given example is passed through each binary classifier, and an aggregation function is applied to the outputs to produce a final prediction. In this paper, the aggregation function selects the highest confidence positive sample prediction (`argmax` across the higher outputs of all ensemble members). This approach is called OVA-NEAT (Figure 1).

The OVA approach is prone to imbalance issues. Each binary classifier is trained using all training data, meaning that even when all N training examples are distributed evenly between the k classes, the decomposed problems are imbalanced, with a negative-to-positive sample ratio of $N - n_i : n_i$, where n_i is the number of positive samples available for task i. Indeed, imbalance increases as the number of classes k increases. An alternative divide and conquer approach does not exacerbate imbalance in this manner.

3.2.2 One-vs-One-NEAT. One-vs-One (OVO) multiclass decomposition [11] eases imbalance and further simplifies the task of each classifier. OVO generates $\frac{k(k-1)}{2}$ binary classifiers, where each classifier discriminates between exactly two classes (i, j) where $i \neq j$. These binary classifiers are once again the champions of individual NEAT runs. At prediction time, a voting scheme is applied: the class with the highest number of positive predictions from the pool of pairwise classifiers is selected (Figure 1). This voting scheme is effective despite the fact that the majority of classifiers are structurally incapable of voting for the correct class for any given sample.

OVO further simplifies the task of each classifier, which must differentiate between only two classes. OVO also mitigates imbalance concerns: balanced multiclass problems remain balanced after decomposition. These advantages come at the expense of additional complexity. OVO generates order k^2 classifiers, compared to k classifiers generated by OVA. However, each OVO classifier is only trained using examples from (i, j), resulting in fewer network evaluations per generation in each subproblem.

These differences in evaluation are taken into account in order to fairly compare these methods in the experiments described next.

Problem	Train	Test	k	A	I
hypercube	$500k \times 0.9$	$500k \times 0.1$	$3, 5, \ldots, 17, 19$	10	N
mnist	60,000	10,000	10	784	N
digits	1,617	180	10	64	N
shuttle	43,500	14,500	7	9	Y
letter	15,000	5,000	26	16	N
cardio	1,613	513	3	48	Y
satellite	4,435	2,000	6	36	N

Table 1: Problem Set Descriptions. Breakdown of characteristics for each experimental problem set: train and test size, classes k, input attributes A, and presence of label imbalance I. These problems represent a diverse set of challenging problems that allow for interesting comparison of various methods.

4 EXPERIMENTAL SETUP

Traditional NEAT is compared to the OVO and OVA ensemble techniques in a variety of synthetic and real-world multiclass classification problems. Since all three approaches depend on NEAT, an open source Python implementation of NEAT[2] is used throughout all experiments. The NEAT hyperparameter configuration used throughout all experiments is described in Appendix A. Manual tuning has indicated that optimal parameter choices vary slightly for different problem sets, but the settings selected for this paper work well with all participating datasets.

Different classification approaches are compared in terms of wall time using the same hardware and underlying NEAT configuration. All experiments are run on a Google Cloud instance with 64 Intel Haswell vCPUs and 240 GB of RAM. Although comparing evolutionary approaches in terms of the number of generations or the number of fitness evaluations is more common in the literature, such a comparison would obscure the fact that evolved networks of different complexity take varying amounts of time to evaluate. Specifically, models evolved by NEAT for multiclass problems are more complex than their binary decomposed counterparts. Models also grow more complex over time, meaning that evaluation is more computationally expensive as generations progress.

To make the comparison between standard NEAT and ensemble approaches as fair as possible, individual runs are allotted one hour to complete. For standard NEAT, a single run simply progresses for as many generations as possible within one hour, though fitness often stagnates short of the allotted time. For ensemble approaches, each component run of NEAT is only allowed to run for one portion of an hour, where portions are determined by evenly dividing the hour by the number of component models to be evolved. To assure fair use of the available hardware resources, these component NEAT runs occur sequentially, so that the total time to complete one ensembling run is still one hour. However, in practice the inherent parallelism of these ensemble approaches allows for much greater efficiency, so the results presented could actually have been achieved in a fraction of the allotted time.

Datasets used in all experiments are multiclass classification problems ($k > 2$) either taken from the literature or generated using open source libraries. Synthetic datasets were generated using the `make_classification` function available in the popular scikit-learn Python package[3], which generates clusters of normally distributed points about the vertices of a hypercube of configurable dimensionality and offers control over problem difficulty through parameters such as the number of informative features, interdependence between features, samples, noise, and classes for a given generated dataset [13]. These datasets are referred to as *synthetic hypercube* datasets throughout the paper. Additionally, a variety of widely used multiclass classification problem sets from the UCI Repository of Machine Learning Datasets were selected to form a more realistic breadth of evaluation [19]. Characteristics of these problem sets are cataloged in Table 1.

5 RESULTS

The results indicate that ensemble approaches are superior to standard NEAT, especially as the number of classes grows, in both synthetic and real-world problems. From this point on, each approach is referred to with Smallcaps (e.g., Neat), and experimental datasets are referred to in **boldface**. Results explore the following themes in order: degradation as the number of classes increases; efficiency of evolution; breadth evaluation across open source datasets; and finally, complexity and behavior of evolved models.

5.1 Multiclass Degradation

Divide and conquer approaches always perform at least as well as Neat, with large gaps in performance developing as the number of classes increases. Figure 2 displays the performance of Neat, Ova-Neat, and Ovo-Neat on tasks featuring a varying number of classes. The tasks were generated using the synthetic hypercube method described in Section 4: classification problems with k target classes are generated for $k \in \{3, 5, \ldots, 17, 19\}$ with 5 informative input attributes, 10 total attributes, 500 samples per class, and a separation distance between hypercube vertices of 1.5. Results are reported only on a holdout test set which was selected randomly using a 90/10 train-test split from the generated data and averaged over ten separate runs. Translucent bands indicate 95% confidence intervals using bootstrap sampling. As the number of classes increases, the performance of Neat rapidly degrades, while Ova-Neat and Ovo-Neat hold steady, dropping less than 0.05 in F1 score. Final F1 scores for each ensemble approach are significantly better than Neat's for $k \geq 7$ (Wilcoxon Rank-Sum Test, $p < 0.05$). For the $k = 19$ case, ensemble F1 scores are about 0.7 greater than NEAT's, or over 400% greater. Furthermore, the translucent bands indicate that Ova-Neat and Ovo-Neat exhibit less variance than Neat, a result consistent with ensemble work using other models [11].

5.2 Evolutionary Efficiency

Although ensemble methods have better final performance as the number of classes increases, it is natural to question how efficient these approaches are. Figure 3 displays the training curves for Neat, Ova-Neat, and Ovo-Neat for one problem instance (where $k = 15$) from Figure 2. Ova-Neat and Ovo-Neat converge to higher performance much more quickly. For most experiments, ensemble methods reached higher quality predictions within the first five

[2] https://github.com/CodeReclaimers/neat-python

[3] http://scikit-learn.org/stable/

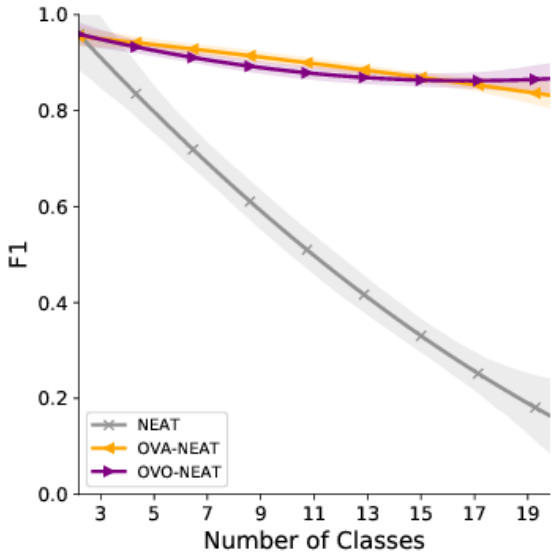

Figure 2: Performance vs. Number of Classes. Average final champion F1 scores across 10 independent one-hour runs reported for validation data as classes increase for synthetic hypercube classification problems. Translucent bands indicate 95% confidence intervals computed using bootstrap sampling.

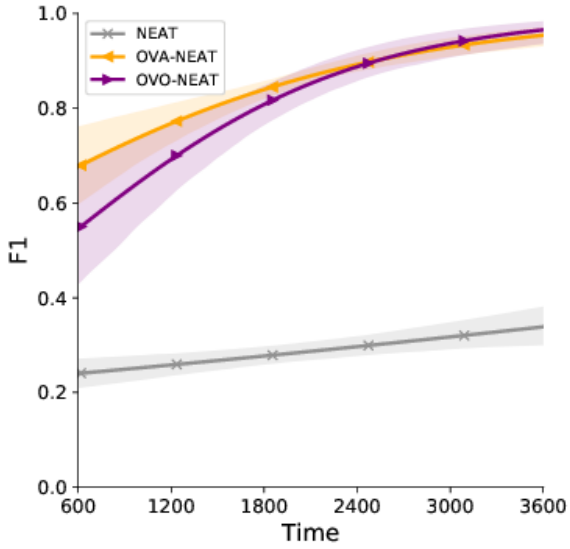

Figure 3: Training Curves. Average champion F1 score across 10 runs on validation data taken at checkpoints every 600 seconds on the synthetic hypercube classification problem for $k = 15$. Translucent bands indicate 95% confidence computed using bootstrap sampling. The 600 second training intervals are evenly divided among constituent models for ensemble methods.

minutes (the first cross-validation checkpoint) than Neat was able to achieve over the course of the entire hour. Furthermore, ensemble methods offer reduced variance at each checkpoint.

Between the ensemble approaches, Ova-Neat tends to produce higher F1 scores overall (Figure 2) and in less time (Figure 3). However, Ovo-Neat is slightly (though not significantly) more robust given sufficient time on problems with a large number of classes. Both of these observations can be understood as consequences of the binarization process for Ovo-Neat, which avoids contributing to data imbalance at the expense of a non-linear increase in the number of component models. Interestingly, even at the rightmost extremity of Figure 2, where $k = 19$ means the number of component models is $(19(19-1))/2 = 171$ and the cumulative evolution time allotted to each model is a mere $3600/171 = 21.05$ seconds, Ovo-Neat still provides the best performance.

Though not reported, results with real data sets were similar.

5.3 Breadth Evaluation

Having characterized the degradation of Neat and its divide and conquer counterparts using an array of synthetic problems, each approach is now applied to a variety of publicly available datasets (Section 4). Table 2 contains evaluation metrics for each of the problems described in Table 1 averaged over ten independent one hour experiments. In each case, metrics are recorded for only the test set. Four metrics are reported: (1) *Accuracy*, the classification accuracy; (2) *F1*, a weighted average of the precision and recall averaged for each class; (3) *W-F1*, the F1 score weighted by the number of samples from each class to account for imbalance; and (4) *Variance* observed in F1 score over the runs.

5.3.1 Performance. Each divide and conquer approach outperforms Neat, with ties only occurring on easier datasets, where Neat already scores quite high. Furthermore, as in Figure 2, the ensemble approaches offer increasingly superior performance as the number of classes increases. For the three problems with the most classes, **mnist**, **digits**, and **letter**, Ova-Neat and Ovo-Neat offer 70-270% improvements in F1 score. In the most extreme case, the 26 class **letter** dataset, there was a 0.463 gap in F1 score between Neat and Ovo-Neat, which is particularly remarkable because Ovo-Neat evolves $26(26-1)/2 = 325$ models for this problem, leaving only $3600/325 = 11.07$ seconds of evolution for each constituent model over the course of each experiment. Additional training time or computational resources result in substantial performance gains for Ovo-Neat on this problem set. In a follow-up experiment (results not in Table 2), Ovo-Neat was restricted to the same one hour training period but allowed to train up to 16 models in parallel, resulting in a final F1 score of 0.91, a 44% increase in F1 score over sequential Ovo-Neat and over 5x the F1 score achieve by Neat. Interestingly, an additional experiment which allotted 16 hours of total training time to Neat (to match the conceivable benefits from parallelism offered to Ovo-Neat) yielded negligible gains in validation accuracy over the added 15 hours of training.

The performance from Ovo-Neat is comparable to results reported by a prior study [16] which applied hand-tuned SVMs aggregated using the OVO and OVA paradigms to a subset of the evaluations here. For example, the 0.999 accuracy score on the **shuttle** dataset exactly matches the numbers reported by SVMs, while the 0.881 and 0.910 accuracies on **satellite** and **cardio** are within 5% of SVM performance.

	Neat				Ovo-Neat				Ova-Neat			
Problem	Accuracy	F1	W-F1	Variance	Accuracy	F1	W-F1	Variance	Accuracy	F1	W-F1	Variance
mnist	0.472	0.367	0.365	0.00381	**0.916**	**0.916**	**0.916**	**0.00008**	0.806	0.801	0.799	0.00009
digits	0.600	0.541	0.508	0.00447	**0.982**	**0.983**	**0.975**	**0.00017**	0.939	0.939	0.939	0.00036
shuttle	0.986	0.985	0.420	0.00010	**0.999**	**0.999**	**0.998**	**0.00004**	0.998	0.931	0.928	0.00016
letter	0.253	0.169	0.165	0.00102	**0.628**	**0.632**	**0.630**	**0.00016**	0.484	0.485	0.454	0.00060
cardio	0.990	0.990	0.986	0.00006	**0.999**	**0.999**	**0.988**	**0.00001**	0.988	0.988	0.986	0.00008
satellite	0.810	0.765	0.708	0.00067	**0.881**	**0.878**	**0.860**	**0.00047**	0.841	0.838	0.809	0.00060

Table 2: Performance Results. Collected and averaged over 10 independent one hour runs for each method and problem set. Ovo-Neat performs best in every category on every data set, as indicated by the bold scores, which are the best in each row. Scores that are underlined indicate results that are significantly better than Neat according to a Wilcoxon Rank-Sum test at the $p < 0.05$ level.

	Neat			Ovo-Neat			Ova-Neat		
Problem	Generations	Nodes	Connections	Generations	Nodes	Connections	Generations	Nodes	Connections
mnist	1,046	94	223	4,790 (106)	1,002 (22)	2,578 (57)	2,877 (288)	337 (34)	817 (82)
digits	2,249	51	148	4,804 (107)	986 (22)	2,717 (60)	3,667 (367)	399 (40)	1073 (107)
shuttle	933	81	205	3,199 (152)	356 (17)	871 (41)	2,607 (372)	264 (38)	647 (92)
letter	1,201	79	182	4,264 (13)	2,313 (7)	14,739 (45)	3,197 (123)	560 (22)	1478 (57)
cardio	967	71	215	1,842 (614)	153 (51)	472 (157)	1,582 (527)	210 (70)	594 (198)
satellite	1,163	72	216	3,807 (254)	465 (31)	1,354 (90)	2,819 (470)	339 (57)	935 (156)

Table 3: Network Complexity. Across 10 runs, the average number of generations to produce champions with an average number of network nodes and links are presented. Ensemble entries are the sums across all constituent runs/champions, but parenthesized values are averaged over these constituent runs/models. Values rounded to nearest whole number.

5.3.2 Imbalance. Ovo-Neat provided the best performance on problems with label imbalance, though both divide and conquer techniques outperformed Neat on such problems. Two of the experimental datasets (**shuttle** and **cardio**) featured heavy class imbalance, with 80% and 78% of examples belonging to a single class, respectively. In the **shuttle** dataset, the remaining samples are split between six classes, whereas in **cardio**, the remaining samples are split between only two classes. Ovo-Neat achieved a considerably higher W-F1 score than Ova-Neat on the **shuttle** dataset (0.998 vs. 0.928), the more difficult of the two imbalanced datasets. This result follows intuition: recall from Section 3 that the binarization procedure of Ova-Neat exacerbates imbalance issues. Interestingly, Neat produces a high accuracy on both imbalanced datasets, but a remarkably low W-F1 score of 0.420 on the 7 class **shuttle** dataset. In contrast, it produces a W-F1 comparable to Ova-Neat and Ovo-Neat on the 3 class **cardio** problem. These results suggest that Neat learns to discriminate between the most common labels (i.e., those which provide the largest fitness boost), but struggles to adapt to minority classes in larger multiclass problems. It is worth mentioning that numerous established mechanisms exist for countering imbalance, such as sampling or re-weighting of the fitness function [20]. This paper purposely neglects such methods in order to characterize the natural behavior of each approach; however, additional experiments have produced performance increases for all three approaches through re-weighting of the fitness function.

5.3.3 Variance. Ensemble approaches also reduce variance across experimental runs. Table 2 displays the variance in F1 score of each dataset. Overall, Ova-Neat exhibited less variance than Neat in 4 out of the 6 experiments, with an 81% average reduction in variance. Ovo-Neat offered the lowest variance across all datasets. These results are consistent with previous research. Specifically, Neat is stochastic and vulnerable to what prior literature has named *the statistical problem*, in which a learning algorithm may independently discover multiple hypotheses which provide similar training accuracy but different generalization capabilities [9]. Decomposition helps ameliorate this risk both by simplifying the search space of hypotheses and entertaining multiple votes from constituent models and thus reducing dependence on any single hypothesis.

5.4 Evolved Complexity

Ova-Neat and Ovo-Neat tend to not only develop more robust solutions, but also develop them more quickly. Table 3 offers further insight into these mechanisms. The total number of generations, as well as a count of the total number of nodes and connections of the evolved champion model(s) for each approach and experiment are shown. For ensemble approaches, the sum across all constituent networks is shown, as well as the average per evolved network (in parenthesis). In each case, Ova-Neat and Ovo-Neat develop substantially more total nodes and connections, reaching a maximum of nearly 30x as many nodes in the case of Ovo-Neat on the **letter** dataset. Despite this richer structure, Ova-Neat and Ovo-Neat are able to accumulate far more generations of total evolutionary innovation in the same amount of time. This is because the average network in the population of any NEAT instance of an ensemble has fewer nodes and connections (parenthesized values in Table 3), which in turn results in less evaluation time per generation.

5.5 Behavioral Analysis

Analysis of Figure 2 and Table 2 suggests that though performance of Neat is stable for smaller multiclass problems, it degrades rapidly

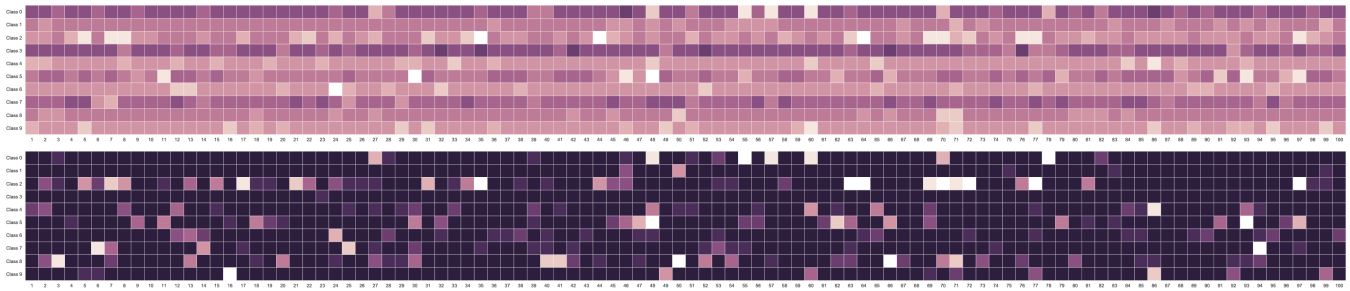

Figure 4: Aggregate Behavioral Heatmaps. Heatmaps of NEAT (top) and OVO-NEAT (bottom) behavior on a synthetic hypercube problem with 10 classes and 100 samples per class. Rows represent different classes. Columns represent samples within a class. A particular cell represents the number of independent runs out of 10 in which the final model correctly classifies the sample, with darker colors indicating a higher number of successful classifications, and white indicating that zero models could classify the sample. For NEAT, some rows are darker than others, but most samples have a pink color for a middling value between 0 and 10. In contrast, all rows are dark in the OVO-NEAT heatmap, indicating that it tends to classify most samples from every class correctly on every run.

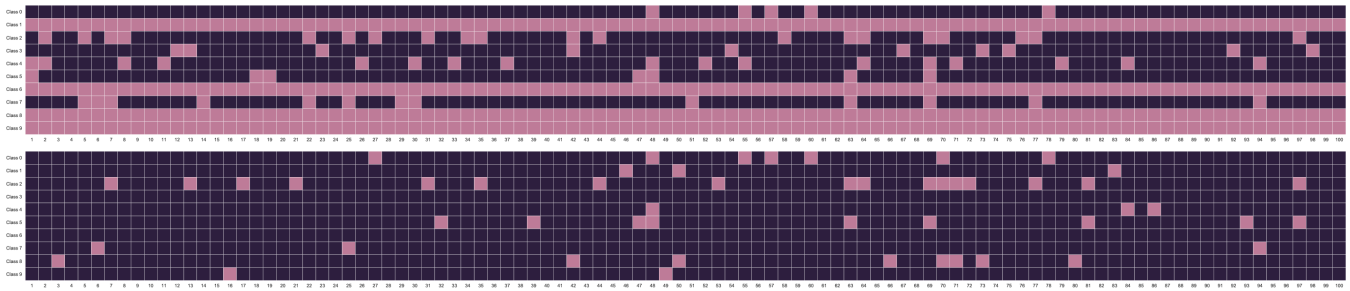

Figure 5: Single-Run Behavioral Heatmaps. Heatmaps of NEAT (top) and OVO-NEAT (bottom) behavior on a single run of the synthetic hypercube problem with 10 classes and 100 samples per class. Samples are organized as in Figure 4, but now there are only two colors: pink (lighter) indicates an incorrect classification, and purple (darker) indicates a correct classification. NEAT correctly classifies samples from a subset of classes, with zero discrimination between the remaining. OVO-NEAT correctly classifies the majority of samples from all classes.

as the number of classes increases, with an inflection point somewhere in the 5–7 class range. The following two figures clarify how NEAT is failing for large k.

Figure 4 is a heatmap visualization of the sample-wise classification accuracy of both NEAT and OVO-NEAT across ten runs of a synthetic hypercube problem (Section 4) with $k = 10$ classes and 100 samples per class. Rows represent different classes, and columns represent samples within a class. Darker shades of purple indicate that more runs successfully classify the sample, while white indicates zero runs correctly classify the sample. The behavior of the two approaches is very different. For NEAT, some rows are darker than others, but the average sample was classified correctly in only 3–6 of the ten runs. In contrast, nearly all samples in all rows are classified correctly in every run by OVO-NEAT.

Figure 5 is the same behavioral matrix produced by only a single run of each approach. Pink (lighter) indicates an incorrect classification, whereas purple (darker) indicates a correct classification. Here, the behavioral differences are clear. NEAT correctly classifies samples from exactly six of the classes, with zero discrimination of the remaining classes, while OVO-NEAT correctly classifies the majority of samples from all classes. This behavior helps explain Figure 4. In a single run, NEAT only evolves topology capable of discriminating between a subset of classes, with some soft limitation between 5–7 classes. The subset of classes captured by NEAT on any particular run is somewhat random, resulting in the mid-range heat of each cell in Figure 4, but some (e.g., "easier" classes or classes with more samples) are more likely to be captured in practice.

6 DISCUSSION AND FUTURE WORK

The results show that divide and conquer ensemble approaches offer a variety of benefits when applying NEAT to multiclass classification problems. The resultant ensembles exhibit reduced variance, superior classification accuracy as the number of classes increases, and improved evolutionary efficiency. This added efficiency derives from the reduced search spaces of the decomposed problems, which admit simpler solutions with fewer nodes and connections, allowing for faster evaluation and richer overall structure. Though standard NEAT can be applied to multiclass problems, it seems better suited to problems with five or fewer classes, making it an ideal candidate for decomposition techniques that have historically been used to extend fundamentally binary classifiers to multiclass problems. Furthermore, divide and conquer approaches introduce a natural point of parallelization that is trivial to leverage in practice. Constituent models may be evolved in different threads, processes, machines, or clusters simultaneously with zero communication between participating nodes. Communication is only necessary when aggregating the decisions of models, enabling enterprise-level applications.

Specifically, Darwin™ is an enterprise automatic machine learning solution which automates data cleaning, pattern discovery, and model creation for data science problems. Darwin™uses a patented combination of genetic and deep learning approaches to converge upon a generalized solution for any supervised, unsupervised, or reinforcement learning problem. The approaches in this paper are among a body of innovations which allow for effective distributed model-building. Though this paper focuses on the simplest instantiation of NEAT, through Darwin™, these ensemble approaches have also been combined with many other enhancements that will be the topic of future research. Specific enhancements include support for deeper architectures, Novelty Search [18], multiobjective optimization [8], the Limited Evaluation Evolutionary Algorithm [21], occasional incorporation of decision trees [5], and application of backpropagation, when appropriate. These various enhancements allow Darwin™to exceed the already impressive performance of pure OVA-NEAT and OVO-NEAT.

With regard to ensemble methods, this paper has shown that ensembles using NEAT are highly effective with a simple decomposition scheme. Future work can pursue further decomposition of binary classification tasks into sub-problems which capture different modes of behavior within classes, which could be useful for complex classification tasks, particularly ones in which a single class consists of multiple unknown/hidden subclasses.

Alternatively, future work might examine complexification as a means of improving classification performance. For example, it is possible that the binarization processes used obscure inter-class information relevant to evolving higher-level structure, such as the information typically encoded in the deeper layers of a deep neural network. If the component networks of an ensemble were periodically merged and evolved/trained further, more complex structure could be discovered, allowing for better discrimination.

7 CONCLUSION

OVA-NEAT and OVO-NEAT are two multiclass classification approaches that combine NEAT with established ensemble methods to attain high performance that greatly surpasses standard NEAT. These techniques are part of the enterprise-level automatic machine learning solution Darwin™, that leverages the highly parallelizable nature of these approaches to quickly solve a variety of data science problems. Future work will analyze many other enhancements to Darwin™, and the ensemble approaches explored here.

REFERENCES

[1] H.A. Abbass. 2003. Pareto Neuro-evolution: Constructing Ensemble of Neural Networks Using Multi-objective Optimization. In *Congress on Evolutionary Computation*. 2074–2080. https://doi.org/10.1109/CEC.2003.1299928
[2] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka. 1995. Efficient classification for multiclass problems using modular neural networks. *IEEE Transactions on Neural Networks* 6, 1 (Jan. 1995), 117–124.
[3] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[4] L. Chen and D. Alahakoon. 2006. NeuroEvolution of Augmenting Topologies with Learning for Data Classification. In *International Conference on Information and Automation*. 367–371. https://doi.org/10.1109/ICINFA.2006.374100
[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Knowledge Discovery and Data Mining*. 785–794.
[6] G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 4 (1989), 303–314.
[7] H. H. Dam, H. A. Abbass, C. Lokan, and X. Yao. 2008. Neural-Based Learning Classifier Systems. *Transactions on Knowledge and Data Engineering* 20, 1 (2008).
[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6 (2002), 182–197.
[9] Thomas G Dietterich. 2002. Ensemble learning. *The handbook of brain theory and neural networks* 2 (2002), 110–125.
[10] Kaibo Duan and S Sathiya Keerthi. 2005. Which Is the Best Multiclass SVM Method? An Empirical Study. *Multiple classifier systems* 3541 (2005), 278–285.
[11] Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* 44, 8 (2011), 1761–1776.
[12] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer. 2005. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE Transactions on Evolutionary Computation* 9, 3 (June 2005), 271–302.
[13] Isabelle Guyon. 2003. Design of experiments of the NIPS 2003 variable selection benchmark. (2003).
[14] Alexander Hagg, Maximilian Mensing, and Alexander Asteroth. 2017. Evolving Parsimonious Networks by Mixing Activation Functions. In *Genetic and Evolutionary Computation Conference*. ACM, New York, NY, USA, 425–432.
[15] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications* 13, 4 (1998), 18–28.
[16] Chih-Wei Hsu and Chih-Jen Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks* 13, 2 (2002).
[17] Nate Kohl and Risto Miikkulainen. 2009. Evolving Neural Networks for Strategic Decision-Making Problems. *Neural Networks, Special issue on Goal-Directed Neural Systems* (2009).
[18] Joel Lehman and Kenneth O. Stanley. 2008. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Artificial Life*. MIT Press.
[19] Christopher J Merz and Patrick M Murphy. 1998. UCI Repository of machine learning databases. (1998).
[20] RA Mollineda, R Alejo, and JM Sotoca. 2007. The class imbalance problem in pattern classification and learning. In *II Congreso Español de Informática*. 978–84.
[21] Gregory Morse and Kenneth O. Stanley. 2016. Simple Evolutionary Optimization Can Rival Stochastic Gradient Descent in Neural Networks. In *Genetic and Evolutionary Computation Conference 2016*. ACM, New York, NY, USA, 477–484.
[22] David Pardoe, Michael Ryoo, and Risto Miikkulainen. 2005. Evolving Neural Network Ensembles for Control Problems. In *Genetic and Evolutionary Computation Conference*. http://nn.cs.utexas.edu/?pardoe:gecco05
[23] Jacob Schrum and Risto Miikkulainen. 2016. Discovering Multimodal Behavior in Ms. Pac-Man through Evolution of Modular Neural Networks. *IEEE Transactions on Computational Intelligence and AI in Games* 8, 1 (2016), 67–81.
[24] Kenneth O. Stanley, Bobby D. Bryant, Igor Karpov, and Risto Miikkulainen. 2006. Real-Time Evolution of Neural Networks in the NERO Video Game. In *National Conference on Artificial Intelligence*. AAAI Press, 1671–1674.
[25] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation* 10 (2002), 99–127.
[26] Xin Yao and Yong Liu. 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* 8, 3 (May 1997), 694–713.
[27] Xin Yao and Yong Liu. 2004. Evolving Neural Network Ensembles by Minimization of Mutual Information. *Hybrid Intelligent Systems* 1 (2004), 12–21.

A NEAT CONFIGURATION

NEAT configuration hyperparameters used for experiments:

```
pop_size = 200; max_fitness_threshold = 1.0;
initial_connection = 0.1; feed_forward = True;
compatibility_disjoint_coefficient = 1.0;
compatibility_weight_coefficient = 0.6;
conn_add_prob = 0.8; conn_delete_prob = 0.1;
node_add_prob = 0.7; node_delete_prob = 0.1;
activation_mutate_rate = 0.3; bias_mutate_rate = 0.7;
response_mutate_rate = 0.0; weight_mutate_rate = 0.8;
compatibility_threshold = 2.5; max_stagnation = 15;
elite_species = 3; survival_threshold = 0.2;
elitism = 2;
```